© INNOVATIVE RESEARCH THOUGHTS | Refereed | Peer Reviewed | Indexed
ISSN : 2454 – 308X | Volume : 07 , Issue : 04 | October - December 2021

DOI : https://doi.org/10.36676/irt.2021-v7i4-31

STUDY OF PSO AND MVO OPTIMIZATION TECHNIQUES FOR TEST EFFORT ESTIMATION

Dr. Vikram Gupta
Associate Professor, PG Department of Computer Science
GSSDGS Khalsa College Patiala
vickysrishti2001@gmail.com

Abstract: It has been suggested that software engineering is a discipline that offers standardised methods for the creation, operation, and maintenance of software. The expenditure of both monetary resources and the man-hours contributed by software Engineers in positions of greater significance are necessary in order to ensure the production of high-quality software. In the software business, determining what percentage of resources are needed during the testing phase may be a key challenge throughout the time of project scheduling. It has been discovered that during the testing phase, around 45 percent of the available resources are needed to be assigned. It is difficult to provide an accurate prediction of the quantity of work that will be required during the testing phase. Finding a mechanism that is able to produce the needed amount of effort throughout the testing phase is the primary focus of the research being done. In order to accomplish effort estimate in an efficient way and with more precision, research is now giving the optimum effort required with the assistance of PSO integrated MVO technique.
Keywords: Test effort estimation, optimization, **Halstead Model,** MVO, PSO,

[1] INTRODUCTION

It is essential to deliver software projects in a timely manner to have an accurate assessment of the amount of work required for software testing. Estimating the testing effort has become a difficult problem because of the complex nature of software projects. However, this is an issue that has to be properly evaluated at the latter stages of the project. The primary causes for incorrect estimates in this field are the presence of insufficient information as well as the lack of clarity on the criteria. In spite of the fact that several models for effort estimating have been suggested over the course of the last decade, the degree of accuracy is not satisfactory enough. In order to arrive at software testing effort estimates that are more precise, the authors of this study describe a novel model that is based on a hybrid mix of PSO and MVO optimization. PSO is an innovative optimization technique that was presented to alter the components of MVO by the use of tiny and appropriate changes in the variables.

1.1 SOFTWARE TESTING EFFORT ESTIMATION

Testing software is a method that identifies and fixes faults that may be present in either the functionality or the code of the programme. The fact that "about thirty-five percent of time and over fifty percent of total expenditure have been spent during testing application" may be taken as an indicator of the significance of testing. The most fundamental objective of testing is to ensure that the application in question is free of errors and that it satisfies both the technical and the business requirements. Executing a system with the goal of finding any faults, bugs, or requirements that are missing from real requirements is what we mean when we talk about testing. Testing is also a means of evaluating and validating an application, which is another way of putting it. When developing an application, one of the most important and management-related activities is called test effort estimation. This activity estimates how long it will take to finish an operation. The following are the primary pillars that comprise an application test effort estimation:
Cost – Every organization wants to earn profits in business. Cost of a application project is budget allocated during application testing. In simple words, it means how much money it takes to finish testing phase. Hence, it's very important to predict perfect budget of application testing life cycle.

Time – Time is key resource in a application project. Every project has a deadline to delivery. So, accurate estimation of time spend in testing process is also an important factor.

© INNOVATIVE RESEARCH THOUGHTS | Refereed | Peer Reviewed | Indexed
ISSN : 2454 – 308X | Volume : 07 , Issue : 04 | October - December 2021

DOI : https://doi.org/10.36676/irt.2021-v7i4-31

Resources – Third important requirement is estimation of resources to carry out application testing of a project. Resources could be people, facilities, equipment or anything else required to complete application testing activity.

Human Skills – It mean knowledge and experience of team members of any application testing project. They could badly affect to your estimation. Let us say, if a team member has less application testing skill, then he / she would take more time to finish his / her work than another member who is expert in technical skills of application testing.

1.2 SOFTWARE TESTING EFFORT ESTIMATION TECHNIQUES

When it comes to forecasting test effort estimates, application developers have access to a wide variety of application test effort prediction approaches. An accurate and reliable assessment of the amount of testing effort might play a key part in large businesses, both in terms of the size of the projects and the sizes of the teams working on them. A strategy for accurately estimating the amount of work required to do testing might be useful to test managers as well as application development projects. The following is a list of main strategies for estimating the testing effort:

Apply the Case Point Method of Estimation

The UCP method is the technique that is considered to be the most essential and is most often used among the many application test effort estimating techniques. Use Cases are the foundation upon which UCP's test effort estimate is built. A use case is essentially an action carried out by the system in its different states. The use case demonstrates how the system reacts to a request from one of the stakeholders under a variety of different circumstances. They have been designated as the major actor. The mapping of use cases to test cases is the most important function of Use Case Point.

Estimation of Effort Based on Function Points and Test Points

During the period of system and acceptability testing, test points are being used for test point analysis in order to determine the amount of work required for testing. Black-box testing is the only kind that is covered by Test Point Analysis. Functional Point Analysis (FPA), on the other hand, does not ever include the system and acceptance test scenarios. As a result, the TPA and FPA are both being blended together in order to compute together the White-Box application testing efforts and the Black-Box application testing efforts. This assists in calculating the amount of time that is necessary for application projects, as well as any risk that may be involved (after comparing TPA and pre anticipated hours). This approach is helpful when measurement of Function Point as well as earlier data for development and testing is available. This is the case while using this method.

Mechanism for Work Breakdown Structure

In the work breakdown structure method, a complicated application project is broken down into separate modules. After then, these modules are broken down into sub-modules based on their function. The functionality of each sub-module is broken down individually. This indicates that the strategy used by this methodology focuses on dividing the overall structure of the application project into smaller structures. There are two distinct varieties of WBS:

When using a functional WBS, the application project is broken down into sections depending on the functionalities of the application that are to be created. This is useful information for determining the size of the system.

In this step of the Work Breakdown Structure (WBS), the application system is dissected based on the activities that make up the system. These actions are then broken down even further into tasks. This is a great exercise for calculating the amount of time and effort required in the application system.

Estimation of the Testing of Software Using Three Points

The three-point estimating approach is one of the most significant effort estimation techniques that might be applied in management and information systems applications. It estimates the amount of work that needs to be done in three different ways. Because of its ease of use, the three-point estimating approach is an extremely helpful instrument for project managers in the process of forecasting the appropriate amount of testing effort to be put into application development projects.

Method of the Delphic Oracle

This is one of the most prevalent methods used for estimating the amount of work required for testing. It's a tried-and-true method for estimating things, and it's based on polls in which information is gathered from specialists about their own previous experiences. In this method of application estimating, each operation is given to a different member of the team, and throughout the course of numerous rounds, surveys are carried out up to the point when a final estimation of the operation has not been determined.

© INNOVATIVE RESEARCH THOUGHTS | Refereed | Peer Reviewed | Indexed
ISSN : 2454 – 308X | Volume : 07 , Issue : 04 | October - December 2021

DOI : https://doi.org/10.36676/irt.2021-v7i4-31

[2] BACKGROUND WORK

Optimized Estimation accuracy is achieved by choosing an appropriate model. Section is providing information of related work is based upon.

2.1 Use Case Point [5]

The use case point analysis will determine the amount of work that has to be estimated for the pre-coding phase [5]. Nageswaran [5] is proposing an approach that calculates effort based on the unadjusted use case weight in addition to the unadjusted actor weight and the technical in addition to the environmental elements. These parameters have been calculated based on the categorization of actors and usecases into simple, average, complicated, and highly complex categories, respectively. To calculate the amount of work required, the acquired unadjusted use case point was multiplied by the factor. During the estimating process, it was discovered that the effort fetch in this was not optimised nor accurate in comparison to the desired precision. The work of Nageswarn has had an impact on the model that has been offered. The findings of this research are contributing to an enhancement of the mechanism proposed by Nageswaran [5]. The Nageswaran model has been described as follows: During this phase of the project, the project manager gets access to the design document, which may be used to determine the amount of work necessary during the testing phase. As a metric for the test effort estimate, the suggested technique suggests using adjusted unadjusted usecase weight, unadjusted actor weight, as well as technical and environmental component. In order to achieve a higher level of accuracy, the hybrid PSO-MVO optimization method was used. According to the design document, the contributors' inputs are gathered for a specific project. The UUCW is determined by using the following use case components:

UUCW = (No. of usecases of type simple*1 + No. of usecases type average*2 + No. of usecases of type complex*3 + No. of usecases of type very complex*4)

The usecase information Table 1 is used for distinguishing and assigning the values. Actor components: The actor information is obtained from the Table 2. TEF components: The technical and environmental factors are assigned as indicated by the Table 3

Table 1 Use case weight assignment table [5]

Usecase type	Description	Weight
Simple	<=3	1
Average	4-7	2
Complex	>7	3

Table 2 Actor weight assignment table [5]

Usecase type	Description	Weight
Simple	GUI	1
Average	Interactive	2
Complex	Low interaction	3

Table 3 TEF weight assignment factor [5]

Factor	Description	Assigned value
F1	Test tools	5
F2	Documented inputs	5
F3	Development environment	2
F4	Test environment	3
F5	Test ware reuse	3
F6	Distributed system	4
F7	Objective	2
F8	Security	4
F9	Complex interface	5

DOI : https://doi.org/10.36676/irt.2021-v7i4-31

UAW and TEF are calculated as:

UAW = ∑ Actor weight*number of actors

TEF = ∑ Assigned Weight*assigned value

2.2 Halstead Model [6]

Past researches has been done by considering the design of Halstead [6]. Here, it is explained in brief. For the purpose of determining software reach as well as its extent some rough measurments are used in this design. [6]. It consider parameters like operator overall strength (n1), operands overall strength (n2), how many operator occur (N1) in addition to how many operand occur (N2). With the help of such measurement a formula for the purpose of determining preliminary work and time required for its work has been put for word by him for the very first time. On the basis of Halstead N is determined in the form of :

N = n1log2 n1 + n2 log2 n2 The program volume is given by the formula

V = N*log2(n1+n2) A volume ratio is defined by him, represented by L, its value should not be more than one. It is represented by the formula L = 2/n1*n2/N2

The effort is given by the formula

Effort = ((n1 * N2)/ (float (2 * n2)) * N *log(n, 2)

This is the effort as estimated by the Halstead model

2.3 Cognitive Complexity [7]

It has been suggested by Kushwaha [7] that it becomes possible to estimate work on the basis of overall weighted data count code and computer programme and fundamental management arrangement. Such type of methods require a difficult determination operator. Its usefulness has not been set up in support of those plans who are larger.

2.4 Effort Estimation Using Soft Computing Techniques [8]

It has been showed by Sandhu [8] that it becomes possible to apply soft computing method like neuronfuzzy for the determination of work. But, for the achievement of this purpose it becomes necessary that its accuracy should be set up by comparing it in the company of additional design. The estimation was done on NASA project data. With the help of Neurofuzzy non linear operations are determined accurately. In actual fact, it has been suggested by this paper that the work which has been estimated on the basis of soft computing are highly accurate .

2.5 HYBRID PSO-MVO

PSO will be used as a kind of evaluation going forward. It takes the shape of a technique that is not too complicated to apply and can be put into practise on a consistent basis. It has already been determined that this kind of evaluation technique finds the best potential answer in the most time-effective way conceivable. This approach may be defined as a procedure that is able to optimise any issue that is taken into consideration within the context of the area of information technology. It has been noted that, in a model that is based on PSO, the efforts put in to improving the performance of candidate solutions are done one at a time. It addresses any and all population-related concerns that are associated with prospective solutions. The so-called particles circulate in and around this area of search space. The calculations for this method are based on an arithmetical procedure that takes into account both the location and the velocity of the particle. Its well-known position within the country has a significant influence on its mobility. To be more specific, it went in the direction of its well-established positions inside the search area. This place now has improved positions according to the latest upgrade. The position of these particles may be readily determined by observing the behaviour of other particles. It is anticipated that this would direct the horde in the direction of the optimal solutions. PSO is considered a met heuristic since it makes very few assumptions, if any at all, regarding the issue that is being optimised. On the other hand, Meta-heuristics like PSO do not provide any assurance that an optimum solution will ever be discovered. In the current scenario, it has become the most essential and valuable met heuristics since, after being used on, it demonstrated success with a variety of optimization issues. It is a model that can arrange itself. It provided a specification about the dynamic nature of these complex systems. In a cooperative and intelligent structure, it uses a highly streamlined model of social behaviour in order to take care of optimization difficulties. This is done in order to make the structure more intelligent.

© INNOVATIVE RESEARCH THOUGHTS | Refereed | Peer Reviewed | Indexed
ISSN : 2454 – 308X | Volume : 07 , Issue : 04 | October - December 2021

DOI : https://doi.org/10.36676/irt.2021-v7i4-31

MVO is a new type of invention. It is an effective maximization method which gets encouragement from environment. Mirjalili et al invented this. For putting this in to operation, two customized factors was kept in mind by them. This method is invented by using three ideology of cosmology. In addition to this form, it also becomes famous in new form of meta-heuristic optimization method. It efficiently figures out those problems which are related to OPF. It is a method which gets continuous motivation from living body & social science stand point. In working of this method different ideology of cosmology are bring in to use. In addition to idea of white & black hole, concept of wormhole is also used in this method. One of most important strong point of this method is that it will find out fast rate of intersection. For this purpose it use roulette wheel selection. In addition to this, this algorithm is able to deal with regular & discrete optimization issues.

Phase 1: In phase one equation of PSO is considered.
Particle Swarm Optimization
Mechanism has been inspired by social expression of birds or fishes. The PSO consists of P_{best} , G_{best}. Position and velocity are updated over course of iteration from these mathematical equations:

$$v_{ij}^{t+1} = wv_{ij}^{t} + C_1R_1(Pbest^t - X^t) + C_2R_2(Gbest^t - X^t) \quad \text{.....(1)}$$

$$X^{t+1} = x^t + v^{2t+1}(i = 1,2..NP)And\ (J = 1,2..NG) \quad \text{.....(2)}$$

Where

$$W = w^{\max} - \frac{(w^{\max} - w^{\min}) * iteration}{\max iteration} \quad \text{.....(3)}$$

w^{max} = 0.4

w^{min} = 0.9. v_{ij}^{t} , v_{ij}^{t+1} has been considered velocity of "j" member of "i" particle in iteration number (t) as well as (t + 1) . (Usually $C_1 = C_2 = 2$), r_1 and r_2 Random number (0, 1).

Phase 2: Multi-verse optimizer equation
Three notions such as black hole, white hole and wormhole are main motivation of MVO algorithm. These three notions are formulated in mathematical models to evaluate exploitation, exploration and local search, respectively. The white hole assumed to be main part to produce universe. Black holes are attracting all due to its tremendous force of gravitation. The wormholes behave as time/space travel channels in which objects could moves rapidly in universe. Main steps uses to universes of MVO:

1. If inflation rate is greater, possibility of presence of white hole is greater.
2. If inflation rate is greater, possibility of presence of black hole is lower.
3. Universes having greater inflation rate are send substances through white holes.
4. Universes having lesser inflation rate are accepting more substances through black holes.

The substances/objects in every universe could create random movement in direction of fittest universe through worm holes irrespective to inflation rate. The objects are move from a universe having higher inflation rate to a universe having lesser inflation rate. It could assure enhancement of average inflation rates of entire cosmoses with iterations. In each iteration, universes are sorted according to their inflation rates and select one from them using roulette wheel as a white hole. The subsequent stages are used for this procedure. Assume that

$$U = \begin{bmatrix} X_1^1 & X_1^2 & . & X_1^d \\ X_2^1 & X_2^2 & . & X_2^d \\ . & . & . & . \\ X_n^1 & X_n^2 & . & X_n^d \end{bmatrix} \quad \text{..... (4)}$$

In equation d is showing the number of variables. The n is showing number of candidate solutions:

$$X_i^j = \begin{cases} X_k^j; & r1 < NI(Ui) \\ X_i^j; & r1 \geq NI(Ui) \end{cases} \quad \text{.....(5)}$$

X_i^j is representing j variable of i universe.
U_i is representing i universe.
NI (U_i) has been considered as normalized inflation rate of i universe.
r1 is a random number from 0 to 1.
X_k^j is showing j variable of k universe selected by a roulette wheel.

DOI : https://doi.org/10.36676/irt.2021-v7i4-31

In order to deliver variations in case of universe and more possibility of growing inflation rate by worm holes, let the worm hole channels have been considered in universe as well as fittest universe made until now. Technique has been formulated as follow:

$$X_i^j = \begin{cases} \begin{cases} X_j + TDR * ((ub_j - lb_j) * r4 + lb_j); \; r3 < 0.5 \\ X_j + TDR * ((ub_j - lb_j) * r4 + lb_j); \; r3 \geq 0.5 \end{cases} ; r2 < WEP \\ X_i^j; r2 \geq WEP \end{cases} \quad \ldots\ldots\ldots(6)$$

X_j is showing j variable of fittest universe created until now.
lb_j is indicating min limit of j parameter.
ub_j is indicating max limit of j parameter.
X_i^j is showing j parameter of i universe.
r2, r3, r4 are random numbers from 0 to 1.
It could be concluded by formulation that wormhole existence probability (WEP) and travelling distance rate (TDR) are chief coefficients. The formula for these coefficients are given by:

$$WEP = min + l * \left(\frac{max - min}{L}\right) \ldots\ldots\ldots(7)$$

Where, l shows present run, and L represent maximum run number/iteration.

$$TDR = 1 - \frac{l^{1/P}}{L^{1/P}} \ldots\ldots\ldots(8)$$

Where, p states accuracy of exploitation with iterations. If p is greater, exploitation is faster and more precise. The complexity of MVO algorithms based on No. of iterations, No. of universes, roulette wheel mechanism, and universe arranging mechanism. The overall computational complexity is as follows:

$$O(MVO) = O(1(O(Quicksort) + n * d * (O(roulette_{wheel})))) \ldots\ldots(9)$$

$$O(MVO) = O(1(n^2 + n * d * \log n)) \ldots\ldots(10)$$

n is showing number of universes
l is showing maximum number of run/ iterations
d is showing number of substances.

Phase 3: Deriving hybrid PSO-MVO equation

Set of Hybrid PSO-MVO is integration of PSO and MVO. Hybrid PSO-MVO is merging best strength of PSO and MVO towards targeted optimum solution. It is replacing PSO Pbest value to MVO Universe value.

$$v_{ij}^{t+1} = wv_{ij}^{t} + C_1 R_1 (Universes^t - X^t) + C_2 R_2 (Gbest^t - X^t) \ldots\ldots(11)$$

3. PROBLEM FORMULATION

At the time of design organization determination of examination work becomes a very difficult task. Up to this point in time, estimation of examination work in an accurate way is almost impossible because already available design fails to do this work optimally. The work done at the examination stage should be determined in an accurate way. It becomes necessary that work should be determined initially ahead of coding stage and after its completion. A design is considered useful in case where comparison results are restricted. Introduction of design by which work can be estimated in an accurate manner becomes the most difficult task. It has been assumed that the design which is introduced here should satisfy project needs.

4. PROPOSED APPROACH

4.1 Architecture

It includes elements like pre and post effort estimation components and optimization mechanism used is PSO and MVO respectively. A comparison has been done in the in the middle of PSO and MVO for the purpose of determining their effectiveness.

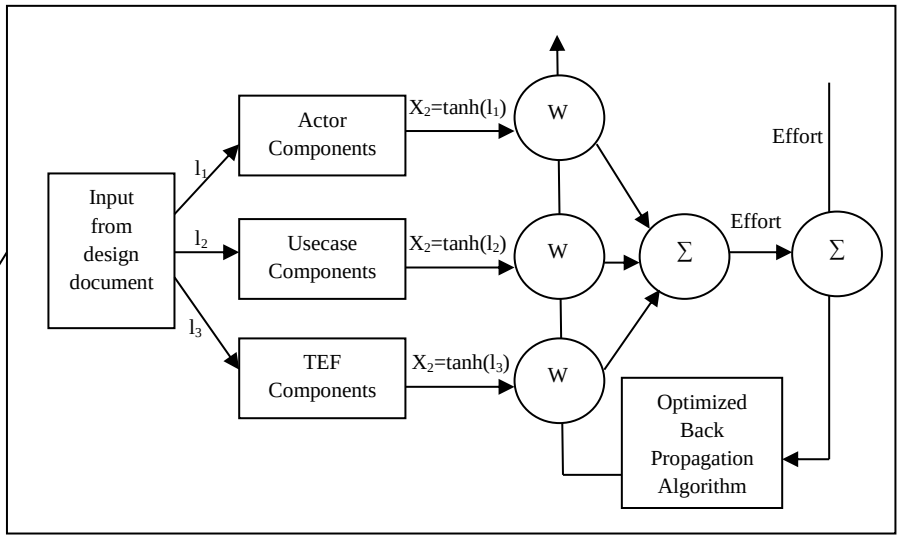
The input components which are used for the purpose of work determination ahead of coding stage get inputs out of model written details and are demonstrated below:

Actor : It gets all the details of those actors which are included within the system.

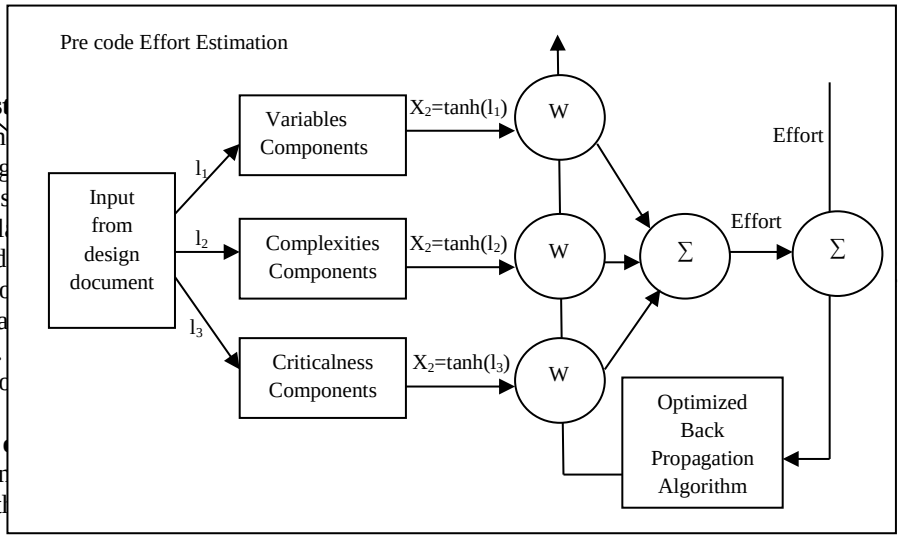
© INNOVATIVE RESEARCH THOUGHTS | Refereed | Peer Reviewed | Indexed
ISSN : 2454 – 308X | Volume : 07 , Issue : 04 | October - December 2021

DOI : https://doi.org/10.36676/irt.2021-v7i4-31

Usecase component: It gets all the details of those use case which are included within the model document.
TEF : It gets all the details of those engineering and atmospheric parameters which are included within the system.
Additional details related to these are already provided in the front of paper.
In a similar way, the input components which are used for the purpose of work determination after the completion of coding stage get inputs out of model written details and are demonstrated below:
Variables component: It gets all the details of those variable which are included within the system
Complexity component: It gets all those details which are related to system complexity. of the system.
Criticalness component: It gets all those details which are related to system criticalness.
Additional details related to these are already provided in the front of paper.
tanh works in the form of activation operator. Represent the inputs given to the system inputs are represented through 'I' whereas 'X' and 'w' has been used for the representation of values achieved post activation operator application and assigned load respectively .



User Project Manger



4.2 se Effort Est
The introduced h
on t swaran desig
On new progress
Not obtained rel
opti of PSO and
and in support
vari that, optima
whi is required.
mechanism provides with the effo

4.3 Optimization of pre coding
During optimization of pre codin
network research is passed to bot

Optimization process: In order to get the optimized pre coding phase effort the optimization function is created. Upper bond and lower bond is set on the bases of dataset. Maximum iterations is also set in order to repeated in various

© INNOVATIVE RESEARCH THOUGHTS | Refereed | Peer Reviewed | Indexed
ISSN : 2454 – 308X | Volume : 07 , Issue : 04 | October - December 2021

DOI : https://doi.org/10.36676/irt.2021-v7i4-31

iterations. The global best and local best are extracted from this simulation. This global best solution supports the pre coding phase effort estimation.

Comparison operation
The accuracy and performance of both optimizers are checked during execution of optimizers. The simulation would conclude which optimizer is best in order to get more accurate result in less time.

4.4 Post coding phase effort estimation
It emerges in the form of phase where the written details related to coding are used by the design manager for the purpose of determining examination work. The method which is introduced here considered that the examination work is entirely depends upon the inputs and outputs number strength , code complexity and its criticalness .
A value is given by the all other important parameters.
Variables component: Examination cases figure is directly proportional to input figure. It means it increment and decrement is entirely depends upon input figure. In support of various inputs number of arrangements is already provided. It becomes possible to observe out of Table four. The method which is introduced here considered that a character data type doesn't need more than single test data, whereas integer information should need additional examination cases and array variable would need further examination cases in support of examination [1]. It is the way in which allocated weights increase in a proportion manner. Values related to the appearance of all varying figure is demanded by var[i] according to the sequence which is given within the Table four. Var_comp[i] exists in the form of appointed load that are demanded out of Table four. In this way, va-riable var_val is obtained by the addition of product of the number of occurrences of variables and their assigned weights.

Complexity component:
The exact test case figure needed for the purpose of research is measured on the basis of code abstruseness. Therefore, a value in support of used code abstruseness is provided in the table five. Appointed load is directly proportional to code abstruseness. It means its increment and decrement is entirely depends upon code abstruseness.

Criticalness component:
Test cases figure is directly proportional to system accuracy. It means its increment and decrement is entirely depends upon system accuracy. It becomes possible to obtain measure out of Table 6. Code accuracy indicates how important code is. A versatile code is designated with lower value (mostly projects are organizes under this category). At the same time, for a useful code examination work is proportional to test cases figure. It increases accordingly in a rapid manner. Therefore, accuracy parameter is is designated with higher value in the form which is shown in Table 6.
sigma () is already specified in the form of variable which acts in the form intermediary variable for the purpose of measuring work. Exactly, achieved from the conclusion of var_val , complexity and accuracy value.

Table 4. Complexity assignment table for variables

Input type	Assigned weight
Integer	3
Array variable	4
Character	1

Table 4. Complexity weight assignment for code

Complexity of the code	Assigned weight
O(n)	1
O(log n)	2
O(nlog n)	3
$O(n^2)$	4
$O(n^3)$	5
$O(n^4)$	6

Table 6 Criticalness assignment table

© INNOVATIVE RESEARCH THOUGHTS | Refereed | Peer Reviewed | Indexed
ISSN : 2454 – 308X | Volume : 07 , Issue : 04 | October - December 2021

DOI : https://doi.org/10.36676/irt.2021-v7i4-31

Criticalness of the code	Assigned weight
General purpose code	3
Higer critical code	2
Mission critical code	1

Var_val =∑ (var[i]*var_comp[i])
α= var_val*complexity*criticalness
Effort = (α+ 13.5)*10/3 (1)

The equation is arrived based on the halstead effort estimation model. The effort is estimated on a large number of test cases (the test cases here being the source codes of quick sort, bubble sort, gcd program etc.,) the halstead effort is estimated for the test cases, the effort is obtained in elementary mental discriminations. For the same test cases the value of α is computed and a large pool of values for the comparison of the proposed variable and the halstead estimated effort is obtained. The constant 13.5 and the multiplying factor 10/3 have been arrived from this large pool of values and their comparisons.

A relation is obtained for the obtained α values and the estimated values. Thus Equation (1) has been derived.
The obtained values var_val and α and estimated best effort according to optimized proposed model. The optimized data is supporting the prediction the values of weights and threshold values for the activation levels. The optimization operation is made through test data over multiple iterations.
Then the model is provided with the information for the project for which an estimate needs to be obtained. The information is derived from the source code document. The various parameters are estimated from the source code like the variable occurrences, complexity of the code etc. The optimizer provides with the effort in terms of the elementary mental discriminations (as the formula was derived using the Halstead model). The optimization operations are performed with the proposed effort estimation function for the post coding phase.

4.5 Optimization of post coding effort estimation

At the time of optimization of post coding effort estimation the data set used for training in previous neural network research is transferred to both PSO and MVO optimizers for simulation operation.

Optimization process: Optimized pre coding phase effort the optimization function is created at the time of optimization. Here lower and upper bond is set along with maximum iteration. Finally global best are calculated after getting local best. These optimized solutions would support the post coding phase effort estimation.

Comparison operation

Again the accuracy and performance of both optimizers are checked during execution of optimizers. The simulation is supposed to confirm which optimizer is best.

5. Application of Proposed Model to Test Cases

The proposed model is providing optimized result to find effort estimation in pre coding phase in person-months and in post coding phase in elementary mental discriminations has been applied to various project data. The data is fetch from Estimator Pal, Usecase point [14] having a detailed design report. This is used in minor as well as major projects. The post estimation model has been found cumbersome. This used to get the proposed optimized value as well as the value that is obtained from Halstead model using PSO and MVO.

6. RESULTS AND DISCUSSION

The simulation has made comparison in case of PSO based and MVO for pre coding effort estimation

MVO optimization technique has been used in order to improve the performance and accuracy during finding optimal solution from precoding and post coding effort estimation. The optimal solution, best objective value and elapsed time

© INNOVATIVE RESEARCH THOUGHTS | Refereed | Peer Reviewed | Indexed
ISSN : 2454 – 308X | Volume : 07 , Issue : 04 | October - December 2021

DOI : https://doi.org/10.36676/irt.2021-v7i4-31

have been considered in order to compare the accuracy and performance of PSO and MVO. The best solution is depending of best objective value.

The results obtained by PSO considering 5000 iteration for pre coding effort estimation is as follow
Optimal solution found in case of PSO is 0
Best objective value 0.3532
Elapsed time is 22.437904 seconds.
The results obtained by MVO considering 5000 iteration for pre coding effort estimation is as follow
The best solution obtained by MVO is: 0.29295
The best optimal value objective function found by MVO is : 0.34989
Elapsed time is 11.626334 seconds.

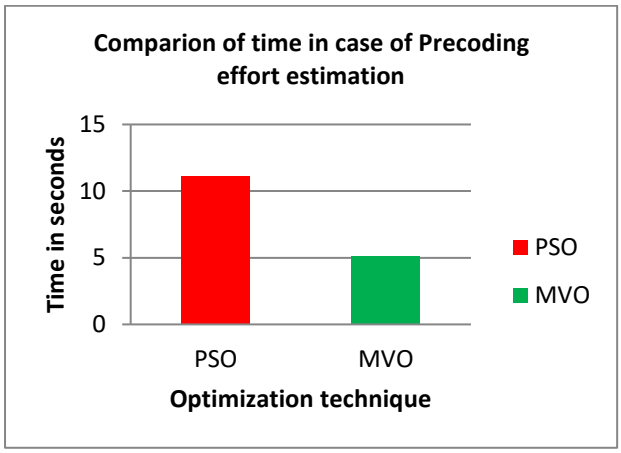


Fig 2 Comparison of time in case of Pre coding effort estimation

COMPARISON IN CASE OF PSO BASED AND MVO BASED POST CODING EFFORT ESTIMATION
The results obtained by PSO considering 5000 iteration for post coding effort estimation is as follow
Optimal solution found is 0.0102
Best objective value 0.3012
Elapsed time is 11.116989 seconds.
The results obtained by MVO considering 5000 iteration for post coding effort estimation is as follow
The best solution for **post coding effort estimation** obtained by MVO is : 0.01025
The best optimal value for **post coding effort estimation** of the objective funciton found by MVO is : 0.30118
Elapsed time is 5.142825 seconds.

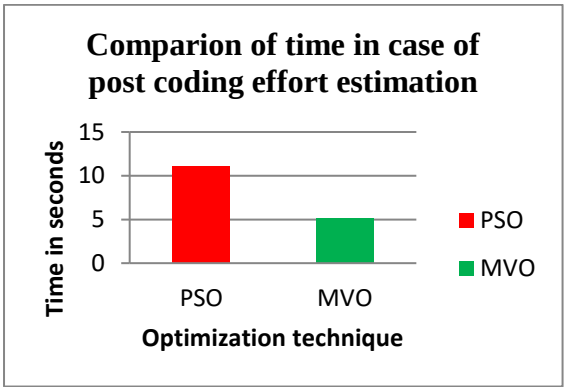


Fig 3 Comparison of time in case of post coding effort estimation

© INNOVATIVE RESEARCH THOUGHTS | Refereed | Peer Reviewed | Indexed
ISSN : 2454 – 308X | Volume : 07 , Issue : 04 | October - December 2021

DOI : https://doi.org/10.36676/irt.2021-v7i4-31

Comparison in case of PSO based and MVO based pre coding effort estimation considering various iterations
Comparison of time in case of PSO based and MVO based **pre coding effort estimation** has been made in following chart. It has been observed that the time consumption in case of MVO based **pre coding effort estimation** is less as compare to PSO based **pre coding effort estimation**. Results are presenting if the iterations are growing then the difference in time consumption is also increasing.

Table 7 Comparison of time for MVO and PSO in case of pre coding effort estimation considering different iterations

Iteration	Pso Based Pre Coding Effort Estimation	Mvo Based Pre Coding Effort Estimation
2000	11.413036	9.85728
3000	16.886031	11.29667
4000	21.597743	14.93652
5000	27.626491	14.03696
6000	36.378653	15.57863
7000	38.607782	17.19762
8000	43.96504	18.5792
9000	52.73117	21.35715
10000	78.413082	21.36492

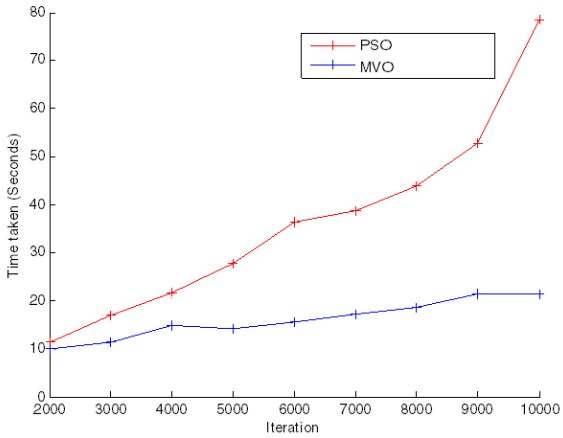


Fig 4 Comparison in case of PSO based and MVO based **pre coding effort estimation**

Comparison in case of PSO based and MVO based post coding effort estimation considering various iterations
Comparison of time in case of PSO based and MVO based **post coding effort estimation** has been made in following chart. It has been observed that the time consumption in case of MVO based **post coding effort estimation** is less as compare to PSO based **post coding effort estimation**. Results are presenting if the iterations are growing then the difference in time consumption is also increasing.

Table 8 Comparison of time for MVO and PSO in case of post coding effort estimation considering different iterations

© INNOVATIVE RESEARCH THOUGHTS | Refereed | Peer Reviewed | Indexed
ISSN : 2454 – 308X | Volume : 07 , Issue : 04 | October - December 2021

DOI : https://doi.org/10.36676/irt.2021-v7i4-31

Iteration	Pso Based Post Coding Effort Estimation	Mvo Based Post Coding Effort Estimation
2000	5.800938	4.552902
3000	8.324576	5.115495
4000	10.799137	5.749414
5000	13.315974	6.14956
6000	15.920454	6.947572
7000	18.550651	7.417791
8000	21.18125	8.055887
9000	23.830014	8.770852
10000	26.661346	9.219504

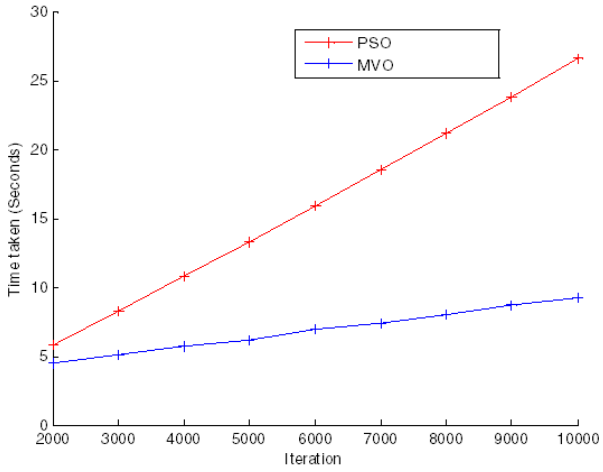


Fig 5 Comparison in case of PSO based and MVO based post coding effort estimation

7. CONCLUSIONS

The use case point or the function point is what is used by the models that are utilised for the more conventional pre-coding effort estimates. In this section, comprehensive information on the procedures that have historically been utilised for the purpose of estimating the amount of work that has been done both before to the beginning of the coding stage and after it has been finished has been presented. After then, many other phrases were included as components of the design that is shown here. The design that has been shown in this article to provide support for before the coding stage functions on the basis of the utilised case point and the way of soft computing. In conjunction with PSO, the use of MVO contributes toward the enhancement of precession. The outcome is provided in an exact way thanks to the followed approach and the supplied metric. The suggested design uses optimization approaches to improve upon

DOI : https://doi.org/10.36676/irt.2021-v7i4-31

accuracy and estimate the work that has been done after the coding step has been completed. This is done in order to estimate the work that has been done after the coding stage has been completed. The outcomes have made it abundantly evident that the suggested estimate is in synchronisation with the firm of normal job determination design. Because the proposed model has the distinct capacity to acquire new knowledge through application, the future application of the model will focus on the direction in which it is necessary for the model that has been developed to be applied to a large number of test cases, also known as real-time projects. Because it has been used for such a long period of time, the design eventually zeroes down on those numbers that are very precise. It is possible to evolve a developed design further in the view that more number of parameters that have a minor effect on the effort estimation be also considered in support of work determination and it is possible to evolve design. This is because more number of parameters have a minor effect on the estimation of the amount of work required.

REFERENCES

[1] R. S. Pressman, "Software Engineering – A Practitioner's Approach," 5th Edition, McGraw Hill, New York, 2002.
[2] B. T. Rao and B. Sameet, "A Novel Neural Network Approach for Software Cost Estimation Using Functional Link Artificial Neural Network," International Journal of Computer Science and Network Security, Vol. 9, No. 6, June 2009, pp. 126-131.
[3] H. Zeng and D. Rine, "Estimation of Software Defects Fix Effort Using Neural Network," IEEE 28th Annual International Computer Software and Applications Conference (COMPSAC'04), Los Alamitos, 28-30 September 2004, Vol. 2, pp. 20-21.
[4] K. K. Agarwal, P. Chandra, et al., "Evaluation of Various Training Algorithms in a Neural Network Model for Software Engineering Applications," ACM SIGSOFT Software Engineering Notes, Vol. 30, No. 4, July 2005, pp. 1-4.
[5] S. Nageswaran, "Test Effort Estimation Using Use Case Points (UCP)," 14th International Software/Internet Quality Week, San Francisco, 29 May-1 June 2001.
[6] T. E. Hastings and A. S. M. Sajeev, "A Vector-Based Approach to Software Size Measurement and Effort Estimation," IEEE Transactions on Software Engineering Vol. 27, No. 4, April 2001, pp. 337-350.
[7] D. S. Kushwaha and A. K. Misra, "Software Test Effort Estimation," ACM SIGSOFT Software Engineering Notes, Vol. 33, No. 3, May 2008.
[8] P. S. Sandhu, P. Bassi and A. S. Brar, "Software Effort Estimation Using Soft Computing Techniques," World Academy of Science, Engineering and Technology, 2008, pp. 488-491.
[9] M. Chemuturi, "Software Estimation Best Practices, Tools & Techniques: A Complete Guide for Software Project Estimators," J. Ross Publishing, Lauderdale, July 2009.
[10] Free Software Foundation, "Neuroph Framework," Version 3, June 2007.
[11] M. Braz and S Vergilio, "Software Effort Estimation Based on Use Cases," 30th Annual International Computer Software and Applications Conference (COMPSAC'06), Chicago, 17-21 September 2006, Vol. 1, pp. 221-228.
[12] G. Banerjee, "Use Case Points – An Estimation Approach," Unpublished, August 2001.
[13] J. Kaur, S. Singh and K. S. Kahlon, "Comparative Analysis of the Software Effort Estimation Models," World Academy of Science, Engineering and Technology, Vol. 46, 2008, pp. 485-487.
[14] N. Nagappan, "Toward a Software Testing and Reliability Early Warning Metric Suite," 26th International Conference on Software Engineering (ICSE'04), Shanghai, 2004, pp. 60-62.
[15] C. Huang, J. Lo, S. Kuo, et al., "Software Reliability Modeling and Cost Estimation Incorporating Test-Effort and Efficiency," 10th International Symposium on Software Reliability Engineering, Boca Raton, 1-4 November 1999, pp. 62-72.
[16] O. Mizuno, E. Shigematsu, Y. Takagi, et al., "On Estimating Testing Effort Needed to Assure Field Quality in Software Development," 13th International Symposium on Software Reliability Engineering (ISSRE'02), Annapolis, 12-15 November 2002, p. 139.